# Writing a Simple Publisher and Subscriber, creating a simple Service and Client, recording and playing back Data, and reading Messages from a Bag File (Python)

Objectives:

1. Learn how to write a simple publisher and subscriber in ROS2 using Python.

2. Implement a basic service and client in ROS2.

3. Record and playback data using ROS2 tools.

4. Read messages from a ROS2 bag file.

Prerequisites:

- Basic understanding of ROS2 concepts.

- ROS2 Foxy Fitzroy or later installed on your system.

- Python 3.x installed.

- ROS2 workspace set up.

Materials Required:

- Computer with ROS2 installed.

- Text editor or IDE.

- Terminal.

Procedure:

In the new terminal, source your main ROS 2 environment as the “underlay”, so you can build the overlay “on top of” it:

source /opt/ros/foxy/setup.bash

Go into the root of your workspace:

cd ~/ros2\_ws

In the root, source your overlay:

source install/local\_setup.bash

From the root of your workspace ros2\_ws, you can now build your packages using the command:

colcon build

1. Writing a Simple Publisher and Subscriber

**Publisher Subscriber:**

C++

[**Create a package**](https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Cpp-Publisher-And-Subscriber.html#id6)

Open a new terminal and [source your ROS 2 installation](https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Configuring-ROS2-Environment.html) so that ros2 commands will work.

Navigate into the ros2\_ws directory created in a [previous tutorial](https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html#new-directory).

Recall that packages should be created in the src directory, not the root of the workspace. So, navigate into ros2\_ws/src, and run the package creation command:

ros2 pkg create --build-type ament\_cmake cpp\_pubsub

Your terminal will return a message verifying the creation of your package cpp\_pubsub and all its necessary files and folders.

Navigate into ros2\_ws/src/cpp\_pubsub/src. Recall that this is the directory in any CMake package where the source files containing executables belong.

[**Write the publisher node**](https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Cpp-Publisher-And-Subscriber.html#id7)

Create a new file named publisher\_member\_function.cpp. Open the file using your preferred text editor.

#include *<chrono>*

#include *<functional>*

#include *<memory>*

#include *<string>*

#include *"rclcpp/rclcpp.hpp"*

#include *"std\_msgs/msg/string.hpp"*

**using** **namespace** **std**::**chrono\_literals**;

*/\* This example creates a subclass of Node and uses std::bind() to register a*

*\* member function as a callback from the timer. \*/*

**class** **MinimalPublisher** : **public** rclcpp::Node

{

**public**:

MinimalPublisher()

: Node("minimal\_publisher"), count\_(0)

{

publisher\_ = **this**->create\_publisher<std\_msgs::msg::String>("topic", 10);

timer\_ = **this**->create\_wall\_timer(

500ms, std::bind(&MinimalPublisher::timer\_callback, **this**));

}

**private**:

void timer\_callback()

{

**auto** message = std\_msgs::msg::String();

message.data = "Hello, world! " + std::to\_string(count\_++);

RCLCPP\_INFO(**this**->get\_logger(), "Publishing: '%s'", message.data.c\_str());

publisher\_->publish(message);

}

rclcpp::TimerBase::SharedPtr timer\_;

rclcpp::Publisher<std\_msgs::msg::String>::SharedPtr publisher\_;

size\_t count\_;

};

int main(int argc, char \* argv[])

{

rclcpp::init(argc, argv);

rclcpp::spin(std::make\_shared<MinimalPublisher>());

rclcpp::shutdown();

**return** 0;

}

#### **Add dependencies**

Navigate one level back to the ros2\_ws/src/cpp\_pubsub directory, where the CMakeLists.txt and package.xml files have been created for you.

Open package.xml with your text editor.

Make sure to fill in the <description>, <maintainer> and <license> tags:

**<description>**Examples of minimal publisher/subscriber using rclcpp**</description>**

**<maintainer** email="you@email.com"**>**Your Name**</maintainer>**

**<license>**Apache License 2.0**</license>**

Add a new line after the ament\_cmake buildtool dependency and paste the following dependencies corresponding to your node’s include statements:

**<depend>**rclcpp**</depend>**

**<depend>**std\_msgs**</depend>**

This declares the package needs rclcpp and std\_msgs when its code is built and executed.

Make sure to save the file.

**CMakeLists.txt**

Now open the CMakeLists.txt file. Below the existing dependency find\_package(ament\_cmake REQUIRED), add the lines:

find\_package(rclcpp REQUIRED)

find\_package(std\_msgs REQUIRED)

After that, add the executable and name it talker so you can run your node using ros2 run:

add\_executable(talker src/publisher\_member\_function.cpp)

ament\_target\_dependencies(talker rclcpp std\_msgs)

Finally, add the install(TARGETS...) section so ros2 run can find your executable:

install(TARGETS

talker

DESTINATION lib/${PROJECT\_NAME})

You could build your package now, source the local setup files, and run it, but let’s create the subscriber node first so you can see the full system at work.

[**Write the subscriber node**](https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Cpp-Publisher-And-Subscriber.html#id8)

Return to ros2\_ws/src/cpp\_pubsub/src to create the next node. Enter the following code in your terminal:

touch subscriber\_member\_function.cpp

Entering ls in the console will now return:

publisher\_member\_function.cpp subscriber\_member\_function.cpp

Open the subscriber\_member\_function.cpp with your text editor.

#include *<memory>*

#include *"rclcpp/rclcpp.hpp"*

#include *"std\_msgs/msg/string.hpp"*

**using** std::placeholders::\_1;

**class** **MinimalSubscriber** : **public** rclcpp::Node

{

**public**:

MinimalSubscriber()

: Node("minimal\_subscriber")

{

subscription\_ = **this**->create\_subscription<std\_msgs::msg::String>(

"topic", 10, std::bind(&MinimalSubscriber::topic\_callback, **this**, \_1));

}

**private**:

void topic\_callback(**const** std\_msgs::msg::String::SharedPtr msg) **const**

{

RCLCPP\_INFO(**this**->get\_logger(), "I heard: '%s'", msg->data.c\_str());

}

rclcpp::Subscription<std\_msgs::msg::String>::SharedPtr subscription\_;

};

int main(int argc, char \* argv[])

{

rclcpp::init(argc, argv);

rclcpp::spin(std::make\_shared<MinimalSubscriber>());

rclcpp::shutdown();

**return** 0;

}

Since this node has the same dependencies as the publisher node, there’s nothing new to add to package.xml.

**CMakeLists.txt**

Reopen CMakeLists.txt and add the executable and target for the subscriber node below the publisher’s entries.

add\_executable(listener src/subscriber\_member\_function.cpp)

ament\_target\_dependencies(listener rclcpp std\_msgs)

install(TARGETS

talker

listener

DESTINATION lib/${PROJECT\_NAME})

Make sure to save the file, and then your pub/sub system should be ready.

[**Build and run**](https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Cpp-Publisher-And-Subscriber.html#id9)

Still in the root of your workspace, ros2\_ws, build your new package:

colcon build --packages-select cpp\_pubsub

Open a new terminal, navigate to ros2\_ws, and source the setup files:

**Result**

Now run the talker node:

ros2 run cpp\_pubsub talker

The terminal should start publishing info messages every 0.5 seconds, like so:

[INFO] [minimal\_publisher]: Publishing: "Hello World: 0"

[INFO] [minimal\_publisher]: Publishing: "Hello World: 1"

[INFO] [minimal\_publisher]: Publishing: "Hello World: 2"

[INFO] [minimal\_publisher]: Publishing: "Hello World: 3"

[INFO] [minimal\_publisher]: Publishing: "Hello World: 4"

Open another terminal, source the setup files from inside ros2\_ws again, and then start the listener node:

ros2 run cpp\_pubsub listener

The listener will start printing messages to the console, starting at whatever message count the publisher is on at that time, like so:

[INFO] [minimal\_subscriber]: I heard: "Hello World: 10"

[INFO] [minimal\_subscriber]: I heard: "Hello World: 11"

[INFO] [minimal\_subscriber]: I heard: "Hello World: 12"

[INFO] [minimal\_subscriber]: I heard: "Hello World: 13"

[INFO] [minimal\_subscriber]: I heard: "Hello World: 14"

Enter Ctrl+C in each terminal to stop the nodes from spinning.